# SYNCHRONIZATION OF SPATIALLY MOVING DELAY COUPLED CHAOTIC OSCILLATORS

**THESIS FOR THE DEGREE OF**

**5 YEARS INTEGRATED M.SC. PHYSICS**

**UNDER THE GUIDANCE OF**

**PROF. BIPLAB GANGULY**


BY

**Bhawik Jani**




Department of Physics

**National Institute Of Technology**
**Rourkela, Rourkela**
**ORISSA-769008**


**May 2014**

**DEDICATED**

To

My Parents

## DECLARATION

I hereby declare that I am the sole author of this thesis. I authorize National Institute of Technology Rourkela to lend this thesis to other institutions or individuals for the purpose of scholarly research.


**DEPARTMENT OF PHYSICS**

**NATIONAL INSTITUTE OF TECHNOLOGY ROURKELA**

**CERTIFICATE**

This is to certify that the project thesis entitled 'SYNCHRONIZATION OF SPATIALLY MOVING DELAY COUPLED CHAOTIC OSCILLATORS' submitted by Bhawik Jani, Integrated M.Sc. student od Department of Physics, National Institute of Technology Rourkela is partial fulfillment for the requirement for the award of degree of Master of Science in Physics and has been carried out by him under my supervision and guidance. The results incorporated in the thesis have been partly reproduced and then extended.

Prof. Biplab Ganguly

Department of Physics

National Institute of Technology Rourkela

Rourkela-769008

## ACKNOWLEDGMENTS

I would like to take this opportunity to express my deep sense of gratitude and profound feeling of admiration to my thesis supervisor. Many thanks to all those who helped me in this work.

# Abstract

Synchronization phenomenon in different systems is analysed in this thesis. First, results of the paper *'From Phase to Lag Synchronization in Coupled Chaotic Oscillators PRL, Michael G. Rosenblum,* Arkady S. Pikovsky, and Jurgen Kurths'* are reproduced and the Fortran program for calculating Leyapunov exponents is given in Appendix 1. In the second part synchronization in an ensemble of oscillators in a finite two dimensional space is studied. The oscillators move in space in a random manner and interaction is governed by their spatial movement and their vision. The coupling interaction or signal travels with a finite speed which leads to delayed coupling and so this becomes a time delayed system. This is an extension of the work done by Lavneet Janagal and P. Parmananda in their paper *'Synchronization in an ensemble of spatially moving oscillators with linear and nonlinear coupling schemes'* in which they studied Synchronization in an ensemble of oscillators with an infinite speed of coupling interaction. Some of their work is reproduced here and a Fortran program is given in Appendix 2. Then the extension to finite coupling interaction speed is described and the respective Fortran code is given in Appendix 3. The major results found by introduction of time delayed coupling are that the extent of synchronization was found to decrease with increasing vision instead of increasing as is the case with infinite coupling interaction speed for very small interaction speed. Also the amplitude of oscillation decreased with decreasing interaction speed.

**TABLE OF CONTENTS**

## Introduction

Synchronization of chaos is a process in which two or more chaotic systems (either equivalent or nonequivalent) adjust a some properties of their motion to a common behavior due to a periodical or noisy coupling. Synchronization of oscillations has been known to scientists since the historical observation of this phenomenon by Huygens in pendulum clocks. With the development of radio and electronics in the 20-th century, synchronization occupied a very special place in science and technology. As many phenomena studied by nonlinear dynamics, synchronization was observed and shown to play an important role in many problems of a most diverse nature (physical, ecological, physiological, meteorological, to name a few). There is hardly a single communication or data storage application that does not rely on synchronization.

Historically, the analysis of synchronization phenomena in the evolution of dynamical systems has been a subject of active investigation since the earlier days of physics. It started in the 17th century with the finding of Huygens that two very weakly coupled pendulum clocks (hanging at the same beam) become synchronized in phase. Other early found examples are the synchronized lightning of fire flies, or the peculiarities of adjacent organ pipes which can almost reduce one another to silence or speak in absolute unison Recently, the search for synchronization has moved to chaotic systems. In this latter framework, the appearance of collective (synchronized) dynamics is, in general, not trivial.

In the first chapter calculation of Lyapunov exponent is discussed following Wolf. In the second chapter, results of the paper *'From Phase to Lag Synchronization in Coupled Chaotic Oscillators PRL, Michael G. Rosenblum,* Arkady S. Pikovsky, and Jurgen Kurths'* are reproduced and the Fortran program for calculating Leyapunov exponents is given in Appendix 1. This describes various types of synchronization and behavior of synchronization with variation of parameters. In the third chapter synchronization in an ensemble of oscillators in a finite two dimensional space is studied. The oscillators move in space in a random manner and interaction is governed by their spatial movement and their vision. The coupling interaction or signal travels with a finite speed which leads to delayed coupling and so this becomes a time delayed system. This is an extension of the work done by Lavneet Janagal and P. Parmananda in their paper *'Synchronization in an ensemble of spatially moving oscillators with linear and nonlinear coupling schemes'* in which they studied Synchronization in an ensemble of oscillators with an infinite speed of coupling interaction. Some of their work is reproduced here and a Fortran program is given in Appendix 2. This is important because there are many phenomenon of nature in which different objects work in synchrony by sharing information or getting information from one source. The objects which synchronize may be moving in space. Then in the

fourth chapter the extension to finite coupling interaction speed is described and the respective Fortran code is given in Appendix 3. This happens when the oscillators interact with a medium which travels with slow speed then light for example sound. The major results found by introduction of time delayed coupling are that the extent of synchronization was found to decrease with increasing vision instead of increasing as is the case with infinite coupling interaction speed for very small interaction speed. Also the amplitude of oscillation decreased with decreasing interaction speed.

## Chapter 1

### Calculation of Lyapunov exponents

In mathematics the Lyapunov exponent or Lyapunov characteristic exponent of a dynamical system is a quantity that characterizes the rate of separation of infinitesimally close trajectories. Quantitatively, two trajectories in phase space with initial separation $\delta Z_0$ diverge (provided that the divergence can be treated within the linearized approximation) at a rate given by

$$|\delta Z(t)| \approx e^{\lambda t}|\delta Z_0|$$

where $\lambda$ is the Lyapunov exponent. The rate of separation can be different for different orientations of initial separation vector. Thus, there is a spectrum of Lyapunov exponents— equal in number to the dimensionality of the phase space. It is common to refer to the largest one as the Maximal Lyapunov exponent (MLE), because it determines a notion of predictability for a dynamical system. A positive MLE is usually taken as an indication that the system is chaotic (provided some other conditions are met, e.g., phase space compactness). Note that an arbitrary initial separation vector will typically contain some component in the direction associated with the MLE, and because of the exponential growth rate, the effect of the other exponents will be obliterated over time. The exponent is named after Aleksandr Lyapunov.

$$\lambda = \lim_{t\to\infty} \lim_{\delta Z_0 \to 0} \frac{1}{t} \ln \frac{|\delta Z(t)|}{|\delta Z_0|}$$

Fortran program to calculate Lyapunov exponents from time series is given in Appendix 1 following *'DETERMINING LYAPUNOV EXPONENTS FROM A TIME SERIES Alan WOLF, Jack B. SWIFT, Harry L. SWINNEY and John A. VASTANO'.*

### Results for Lorenz error equation Lyapunov exponents vs coupling strength

Lorenz system can be written as

$$\begin{aligned}
\frac{\mathrm{d}x}{\mathrm{d}t} &= \sigma(y - x), \\
\frac{\mathrm{d}y}{\mathrm{d}t} &= x(\rho - z) - y, \\
\frac{\mathrm{d}z}{\mathrm{d}t} &= xy - \beta z.
\end{aligned}$$

Here σ, ρ, β are parameters.

Plot of LE of Lorenz error equation is given below as a function of coupling strength with
σ = 10, β = 8/3, ρ = 28
k = 10

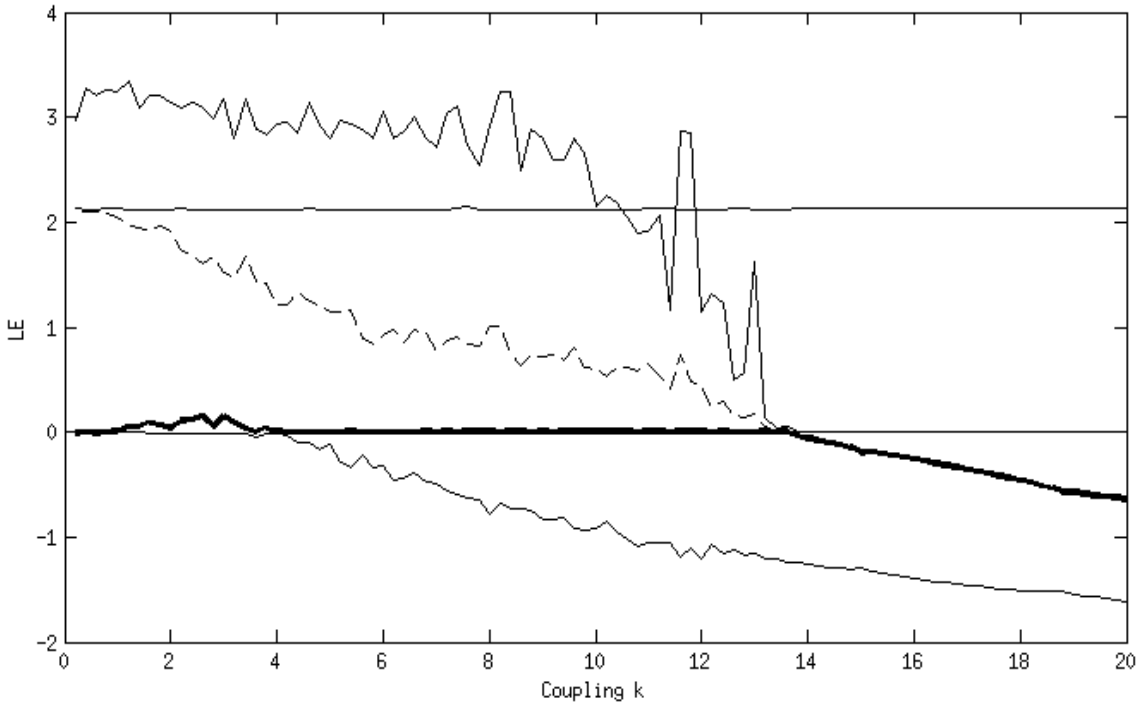



*Figure 1*

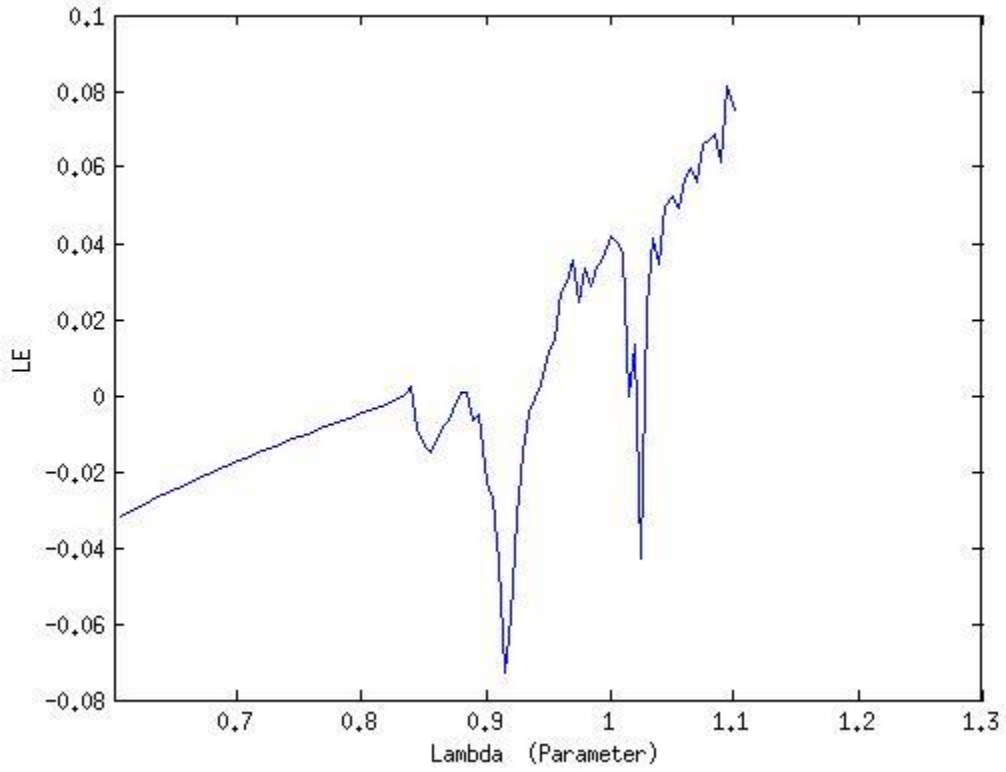
We see that only after the coupling exceeds a certain critical coupling (in this case between 13 and 14) the system begins to synchronize. This can be seen as the error ($X_2$-$X_1$), the top most curve goes to zero and at the same time one positive LE of the error equations becomes zero and the zero LE becomes negative.

**Lyapunov exponent of Van der Pol oscillator vs damping parameter**

In dynamics, the Van der Pol oscillator is a non-conservative oscillator with non-linear damping. It evolves in time according to the second order differential equation:

$$\dot{x} = \mu \left( x - \frac{1}{3}x^3 - y \right)$$

$$\dot{y} = \frac{1}{\mu}x$$




where $x$ is the position coordinate — which is a function of the time $t$, and $\mu$ is a scalar parameter indicating the nonlinearity and the strength of the damping. LE plot is given in figure 2.

*Figure 2. Plot of Van der Pol LE vs the parameter*

## Chapter 2

**Phase and Lag synchronization**

In the paper *'From Phase to Lag Synchronization in Coupled Chaotic Oscillators, Michael G. Rosenblum, Arkady S. Pikovsky, and Jürgen Kurths'* , Lag synchronization was discovered for the first time. It comes as a transition from phase synchronization when there is a parameter mis-match between the coupled oscillators with increasing coupling strength. The results are reproduced here.
Coupled Rossler oscillator is given as :

$$\dot{x}_{1,2} = -\omega_{1,2} y_{1,2} - z_{1,2} + \varepsilon(x_{2,1} - x_{1,2}),$$
$$\dot{y}_{1,2} = \omega_{1,2} x_{1,2} + a y_{1,2},$$
$$\dot{z}_{1,2} = f + z_{1,2}(x_{1,2} - c),$$

where $a = 0.165$, $f = 0.2$, *and* $c = 10$. The parameters $\omega_{1,2} = \omega_0 \pm \Delta$ and determine the mismatch of natural frequencies and the coupling, respectively.

To describe the phase and the lag synchronization, we need to introduce the defination of phase for the Rossler attractor. The phase and the amplitude can be conveniently introduced as :

$$\phi = \arctan \frac{y}{x},$$
$$A = (x^2 + y^2)^{1/2}$$

To characterize Lag synchronization, they introduce a similarity function $S$ as a time averaged difference between the variables $x_1$ and $x_2$ (with mean values being subtracted) taken with the time shift $\tau$ .

$$S^2(\tau) = \frac{\langle [x_2(t + \tau) - x_1(t)]^2 \rangle}{[\langle x_1^2(t) \rangle \langle x_2^2(t) \rangle]^{1/2}},$$

Below are the plots of frequency difference vs coupling k in figure 3 and minimum of similarity function $\sigma$ vs $k$ in figure 4.

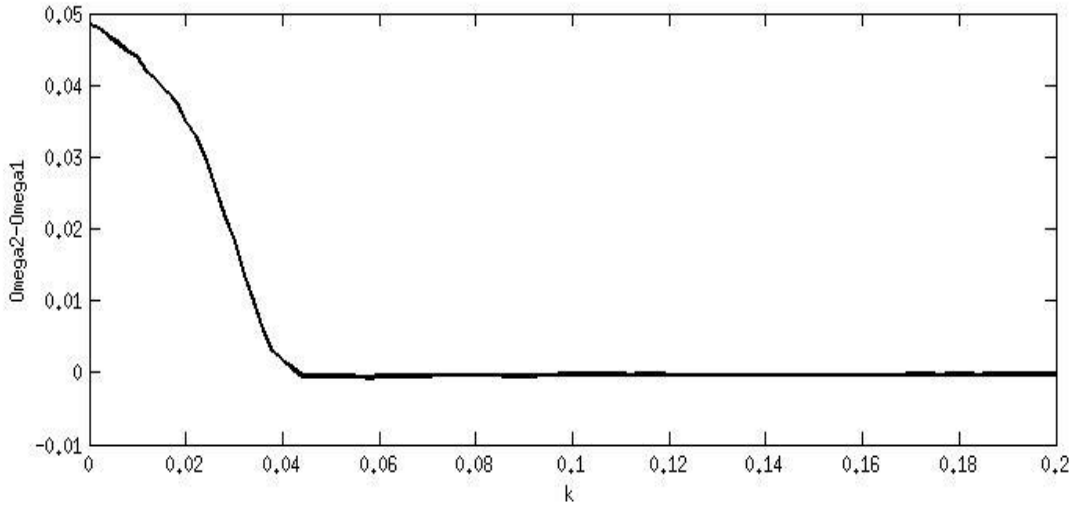



*Figure 3. Frequency difference vs k*

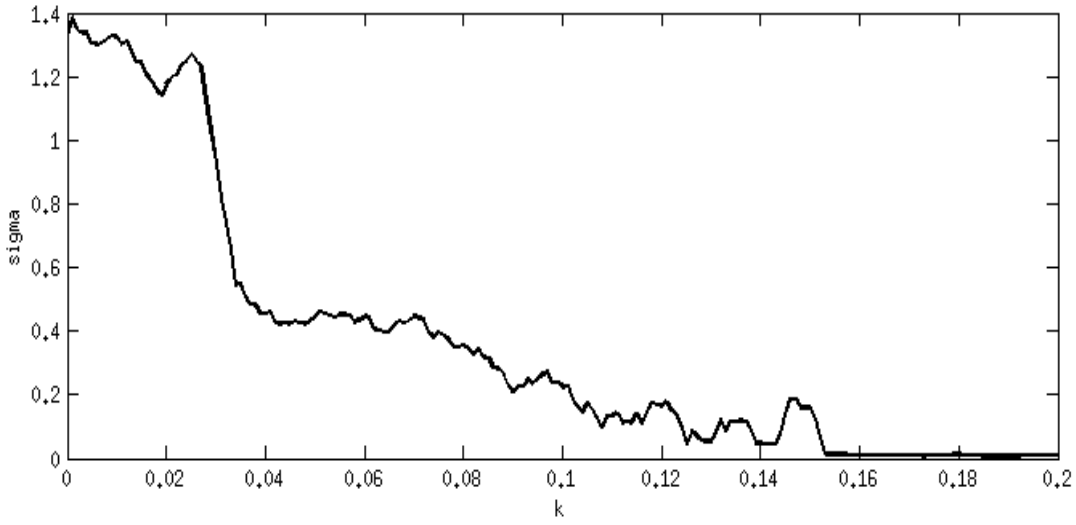



*Figure 4. Sigma vs k*

We see that at $k = 0.04$ there is a transition and phase synchronization occurs as the frequency difference goes to zero.

For stronger coupling $k = 0.14$ they showed a new transition to lag synchronization (see the $\sigma$ vs $\tau$ curve in Fig 5). In Fig. below is shown numerically obtained similarity functions for relatively weak, intermediate, and strong coupling.

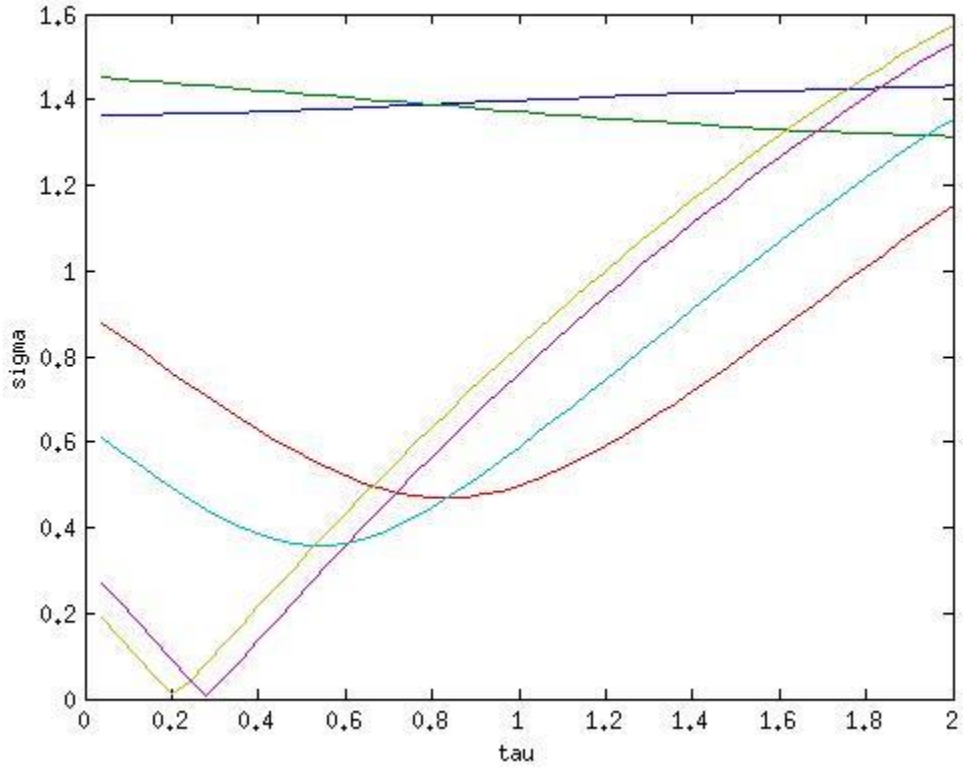



*Figure 5. Sigma vs tao*

We see that for stronger coupling amplitudes are fully co-related but there is a phase lag. This is called Lag synchronization.

## Chapter 3

### Synchronization in an ensemble of oscillators in a finite two dimensional space

Synchronization in an ensemble of oscillators in a finite two dimensional space is studied. Here the work done by Lavneet Janagal and P. Parmananda in their paper *'Synchronization in an ensemble of spatially moving oscillators with linear and nonlinear coupling schemes'* is partly reproduced. In their paper they studied Synchronization in an ensemble of oscillators moving randomly in space. Each oscillator has a given vision and it can couple to all those oscillators which are in its vision but with an infinite speed of coupling interaction. This means no matter how far the oscillators are the coupling travels instantaneously. Some of their results are reproduced in this chapter. In the next chapter I have described the extension of their work that I did to the case where the oscillators move randomly in 2D space but the coupling interaction between oscillators in vision travelling with a finite speed *c.*
In their paper they used Van der Pol and Brusselator oscillators as model system.

Algorithm for movement in space:

1. Take P x Q grid in space.
2. Randomly place N oscillators in P x Q grid.
3. With a probability of 0.25 move the oscillators up, down, left, right by a unit *u* which is the velocity of movement.
4. Use periodic boundary for movement.
5. Update the time and repeat.
6. If the vision is *v* then if an oscillator moves in a given direction, then *v* is the vision in that direction and *0.4v* in the perpendicular direction.

Model for oscillator's internal motion:

Van der Pol oscillator is used for internal motion which is given below

$$\frac{dX_i}{dt} = \mu_i\left(X_i - X_i^3/3 - Y_i\right) + \frac{\lambda}{m_i(t)}[(X_j - X_i) + (X_k - X_i) + \cdots],$$

$$\frac{dY_i}{dt} = X_i/\mu_i,$$

Here $X_i$, $Y_i$ are the variables of the $i^{th}$ Van der Pol oscillator, $\mu$ is a parameter related to damping, and $\lambda$ is the coupling constant. $m_i(t)$ is the number of oscillators with which the $i^{th}$ oscillator is interacting at time t, and j,k,... correspond to the label for the oscillators with which it is interacting.
Fortran program given in Appendix 2.

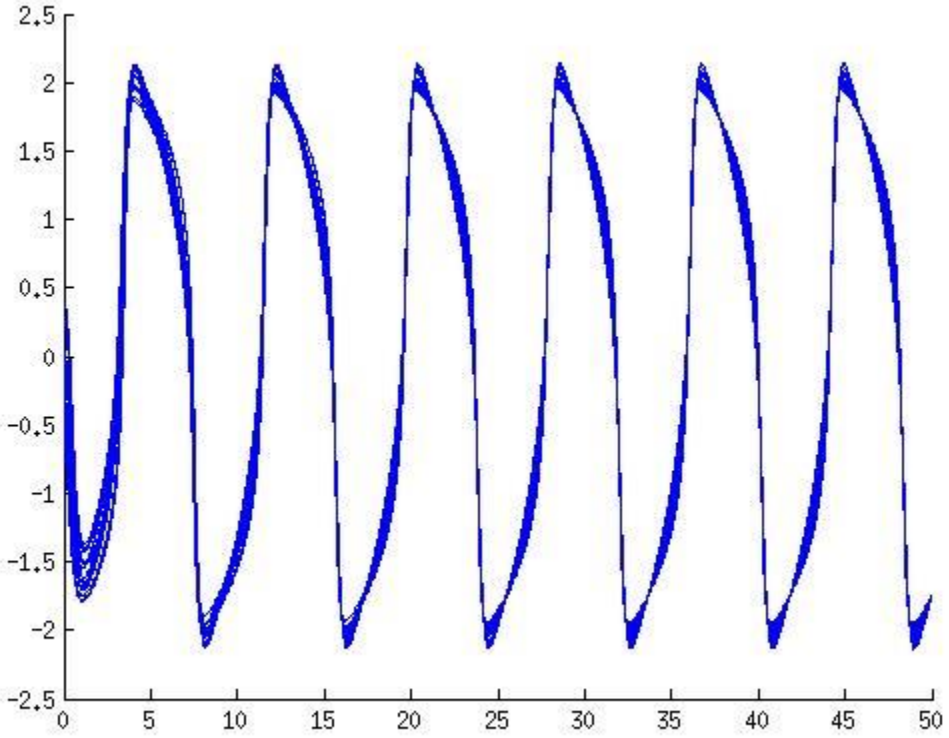
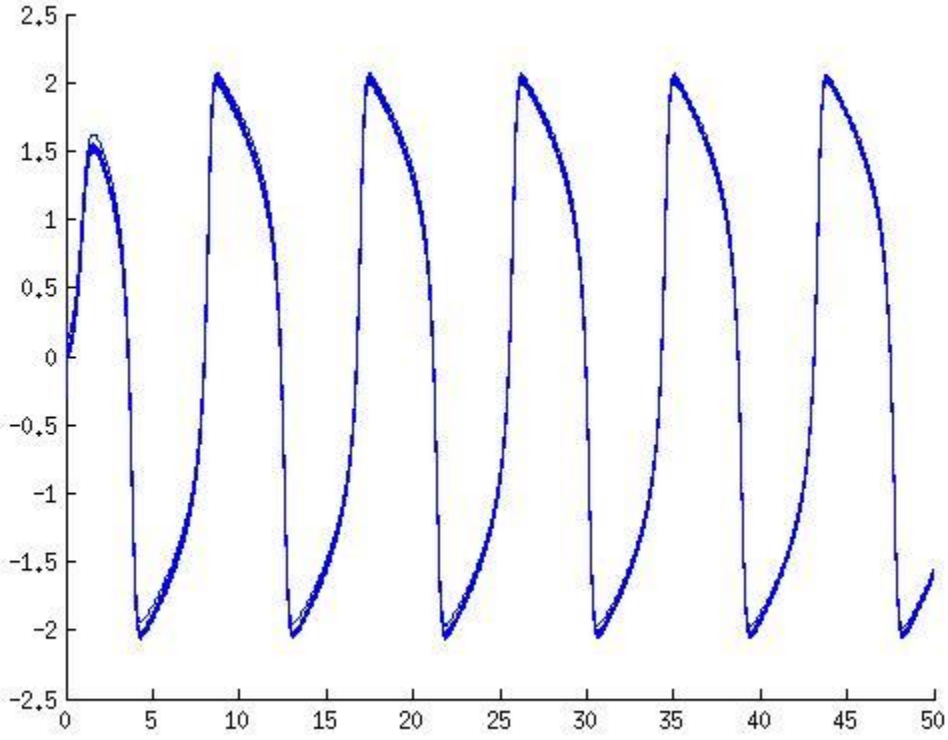


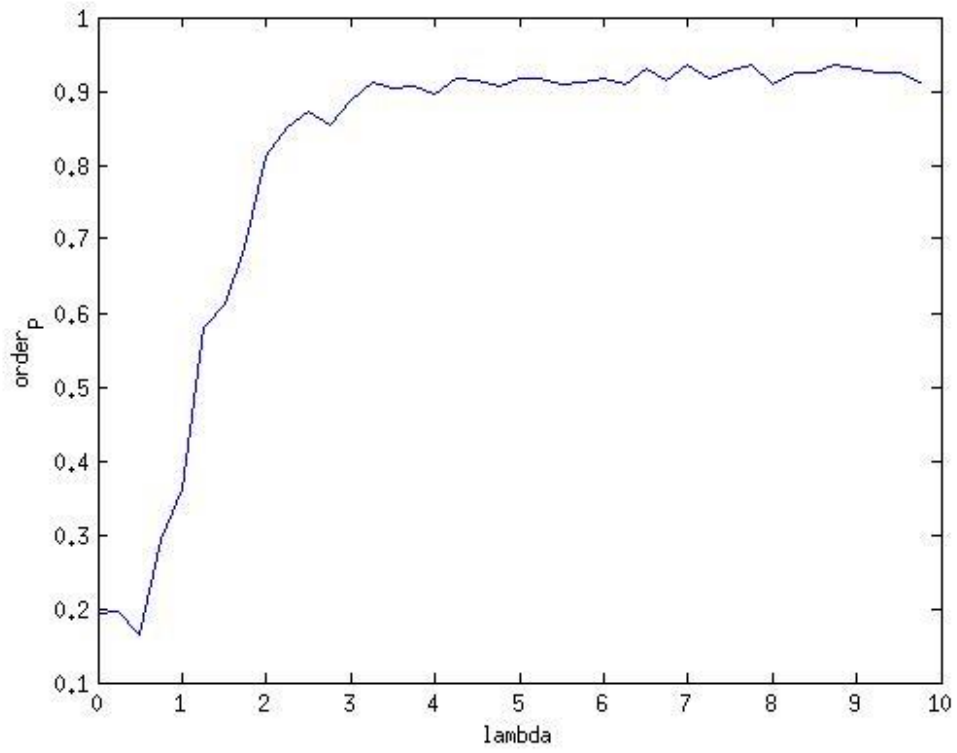

*Figure 6. Superimposed $X_i$ vs time for small coupling*

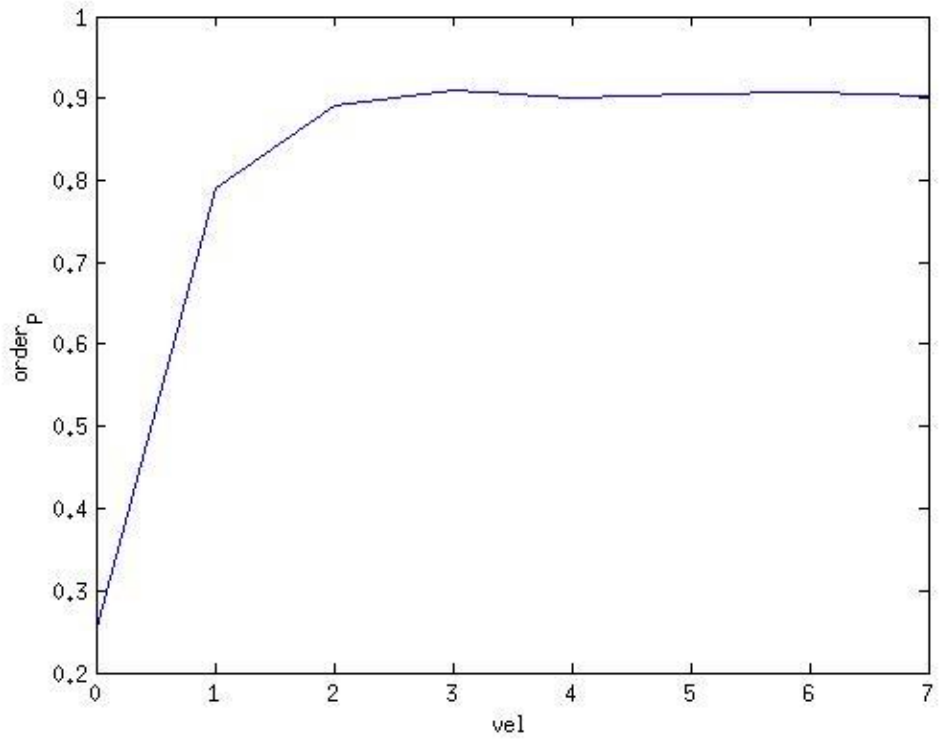
Fig 7

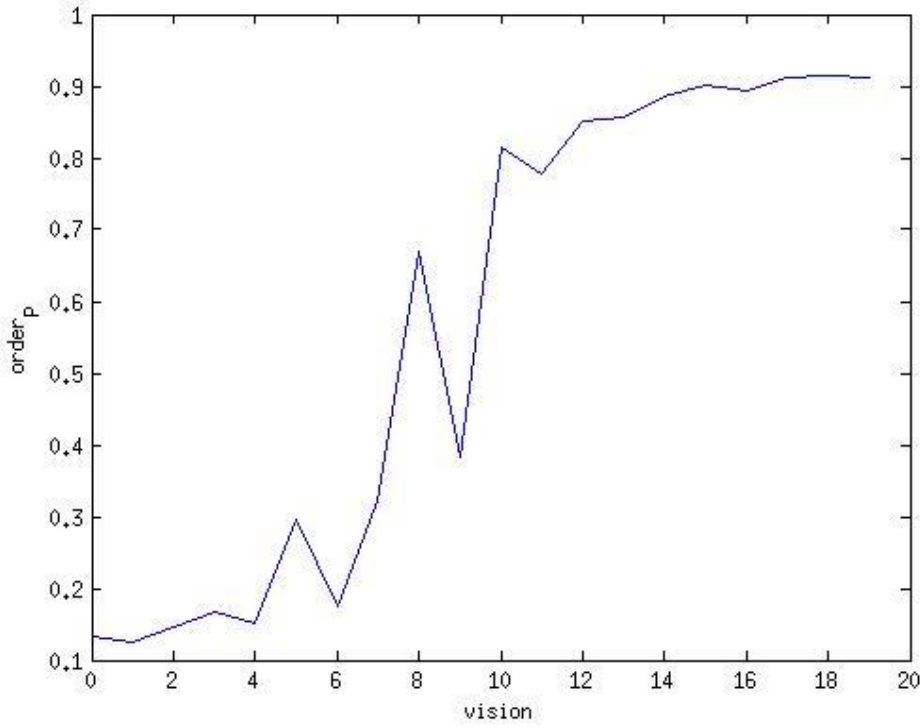
Large coup



Order Parameter is plotted for various values of parameters below






*Figures- 8, 9, 10*

The results found by them by above simulation is that the extent of synchronization quantified by the *Order Parameter* increases monotonically with vision, velocity and coupling constant.

## Chapter 4

### Extension of the previous problem to the case with finite speed of coupling interaction.

I considered the extension of the above problem to the case where the oscillators move in a 2D space randomly with a given vision. The difference here is that the interaction between the oscillators in vision takes place with a finite speed *c*.
This introduces a delay in coupling and makes this a time delayed system

$$\frac{dX_i}{dt} = \mu_i\left(X_i - X_i^3/3 - Y_i\right) + \frac{\lambda}{m_i(t)}[(X_j(\text{t - tao}) - X_i) + (X_k(\text{t - tao}) - X_i) + \cdots],$$
$$\frac{dY_i}{dt} = X_i/\mu_i,$$

Where, *tao* or $\tau$ is the delay introduced by the finite speed *c* of interaction.

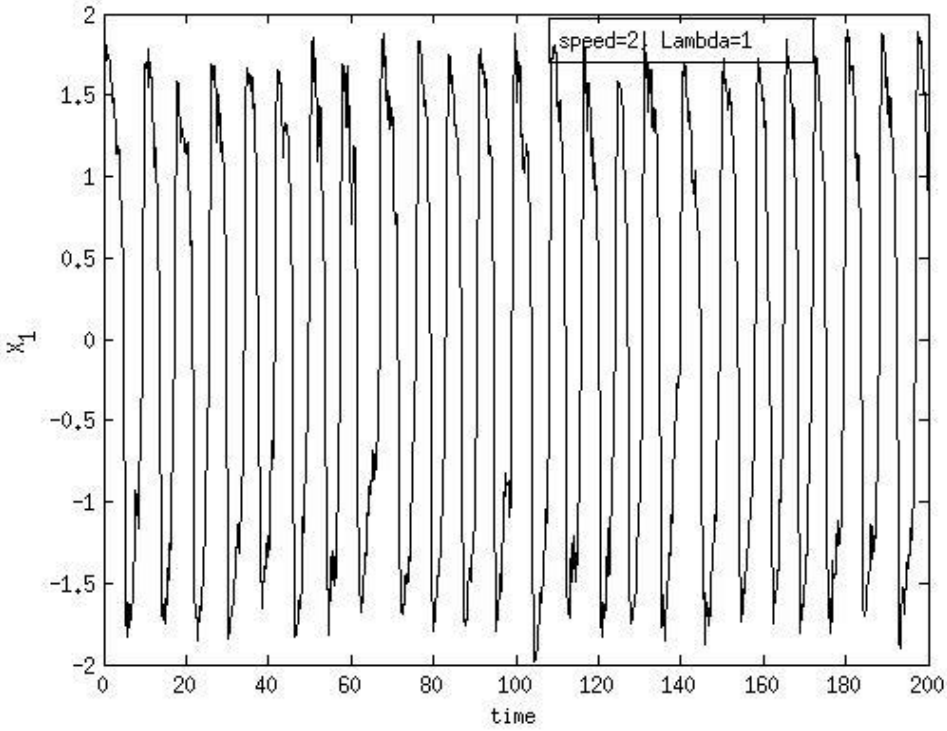
Algorithm for calculating $\tau$

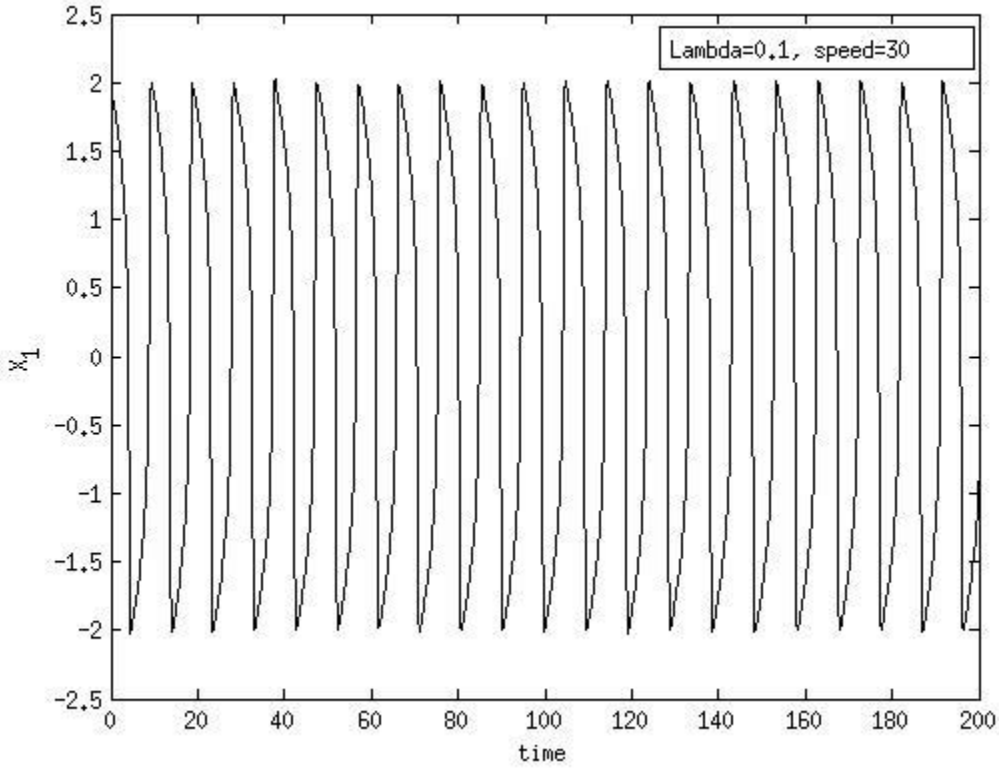
1. Find oscillators in vision.
2. Calculate the distance between the oscillators by formulae
3. $$d=sqrt[(x_2-x_1)^2+(y_2-y_1)^2].$$
4. Calculate $\tau$ by the following

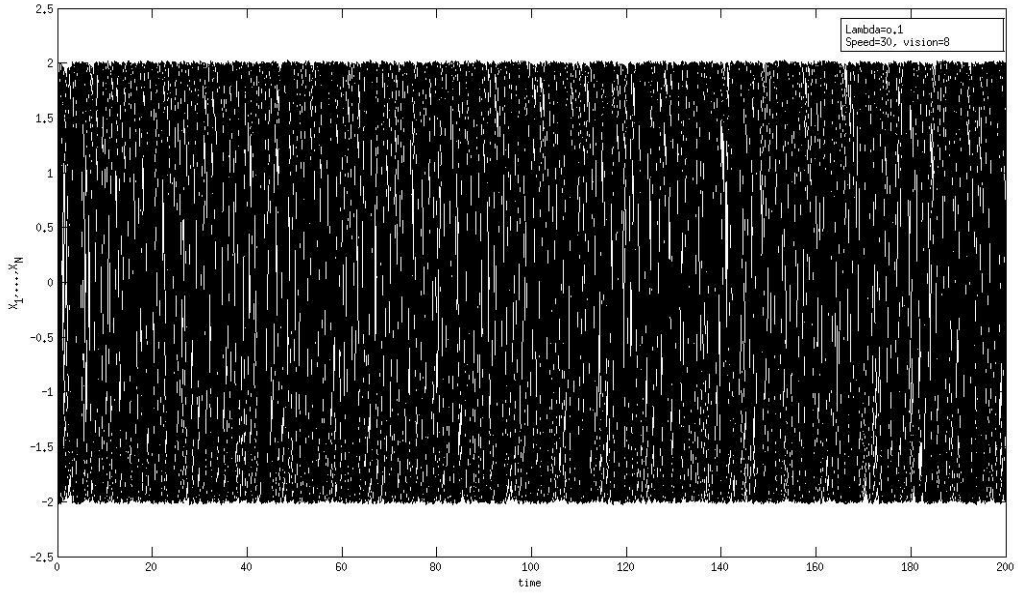
$$\tau = d/c$$

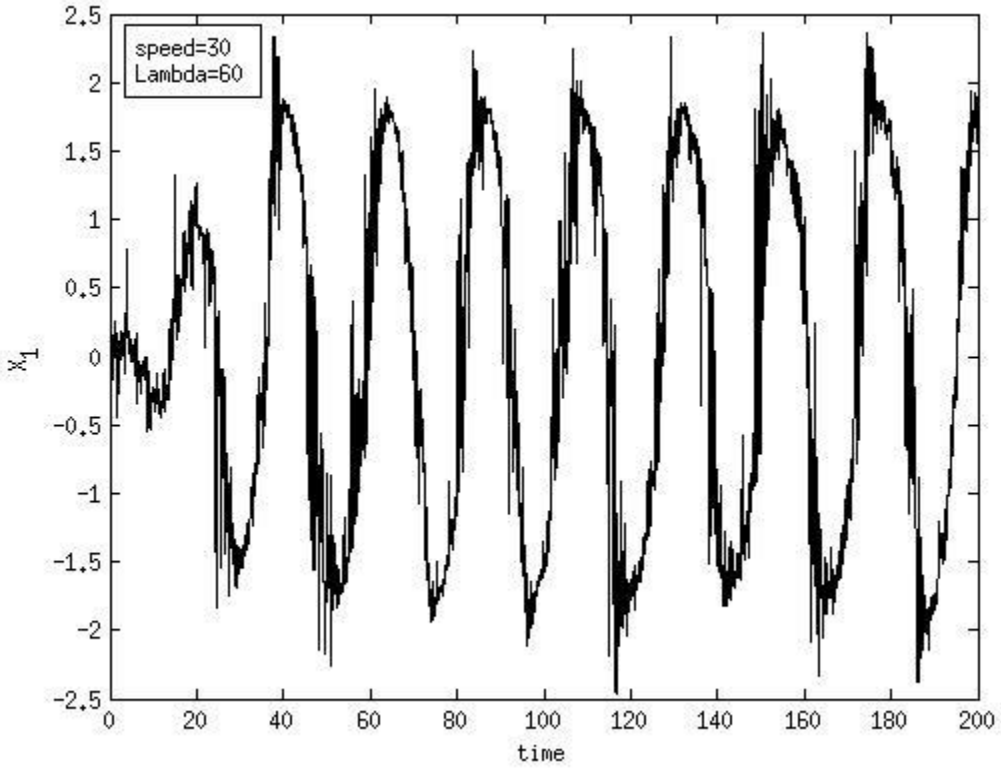
Fortran program for this algorithm is given in Appendix 3

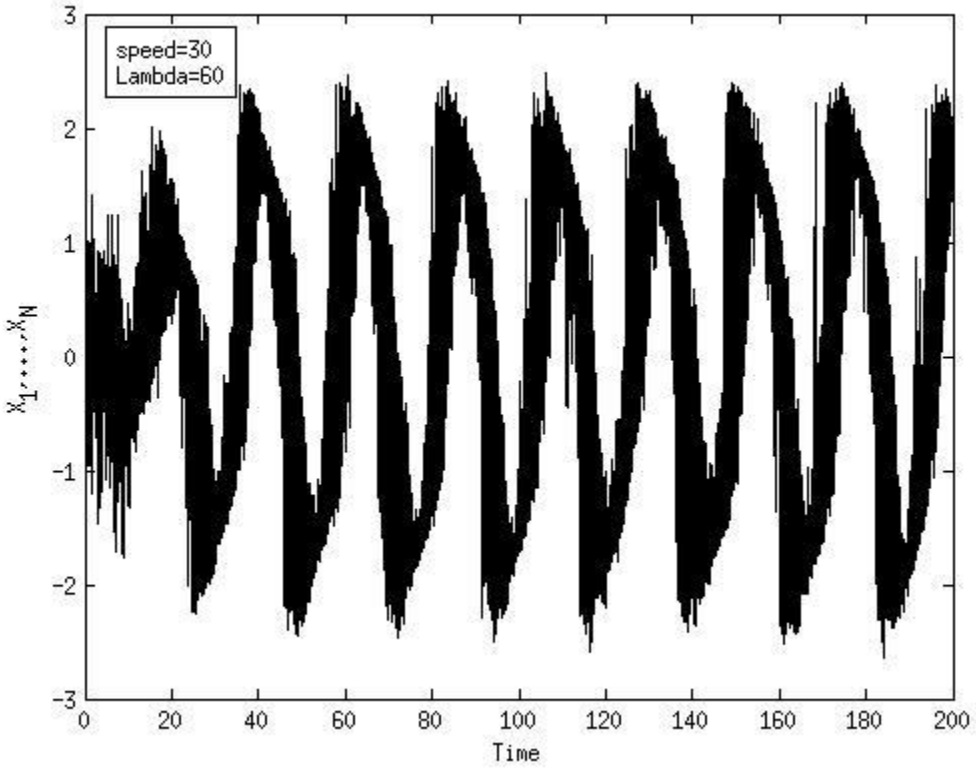
Various results are given below

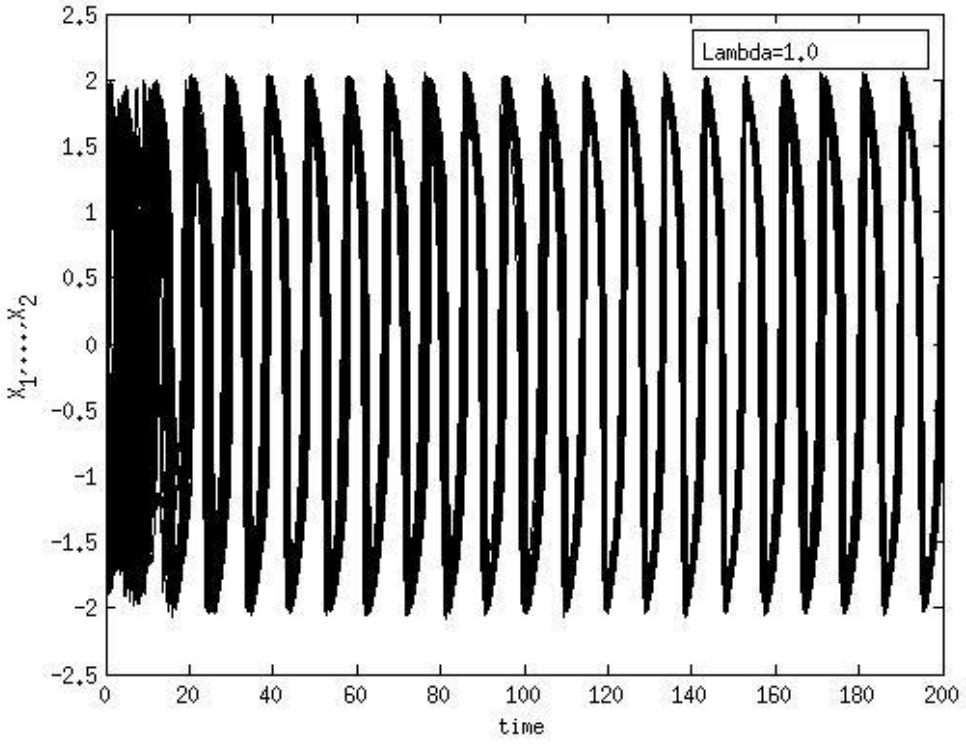

speed=2, Lambda=1
x1
time
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
20
40
60
80
100
120
140
160
180
200

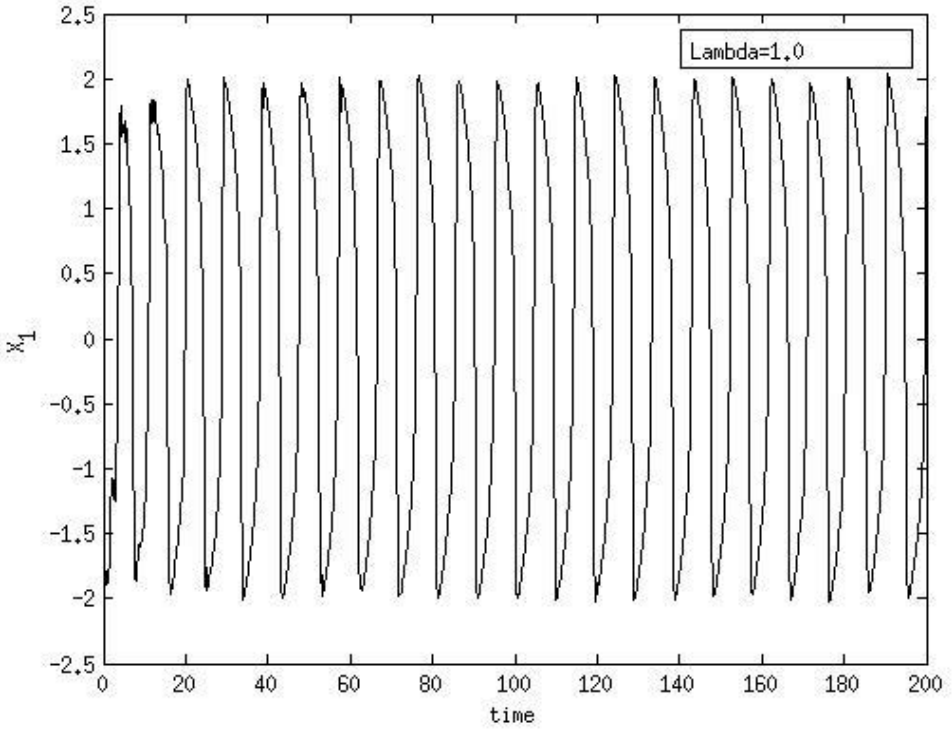

Lambda=0.1, speed=30
x1
time
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
20
40
60
80
100
120
140
160
180
200

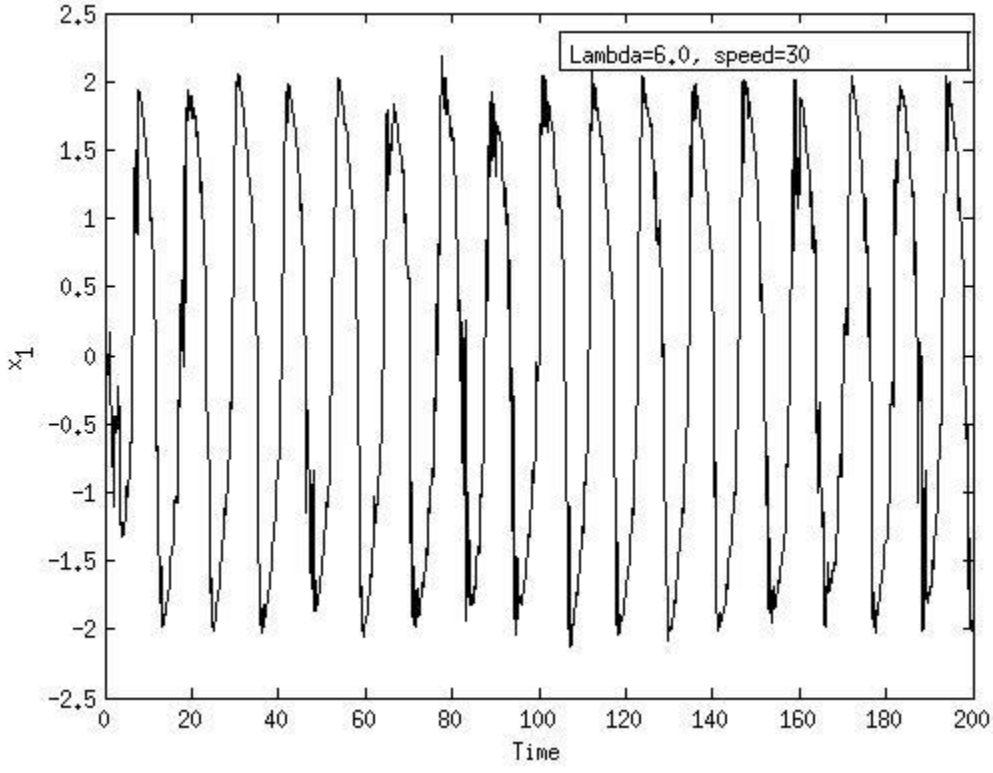

Lambda=o.1
Speed=30, vision=8
time

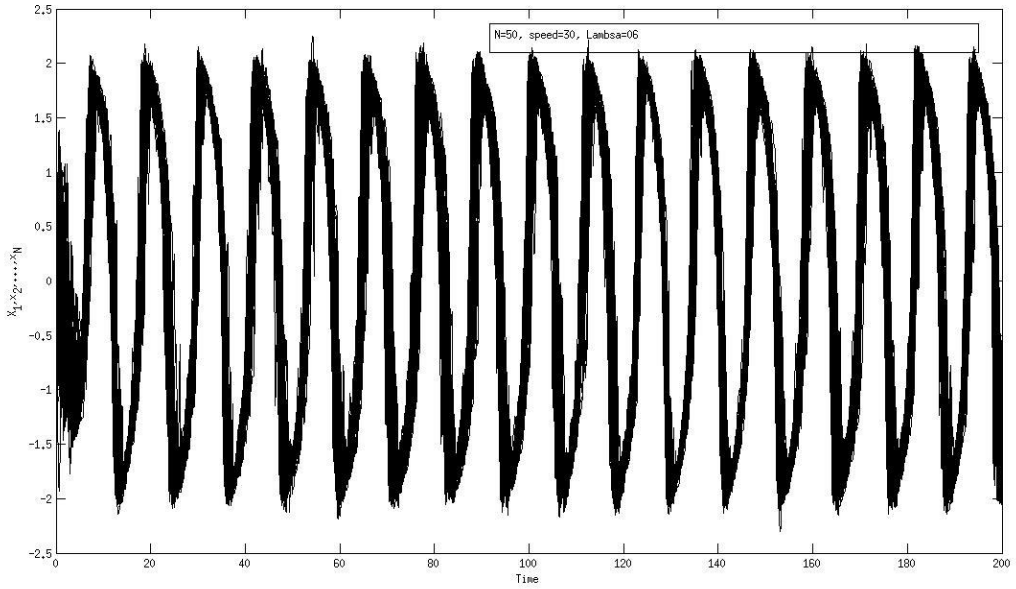

speed=30
Lambda=60
time

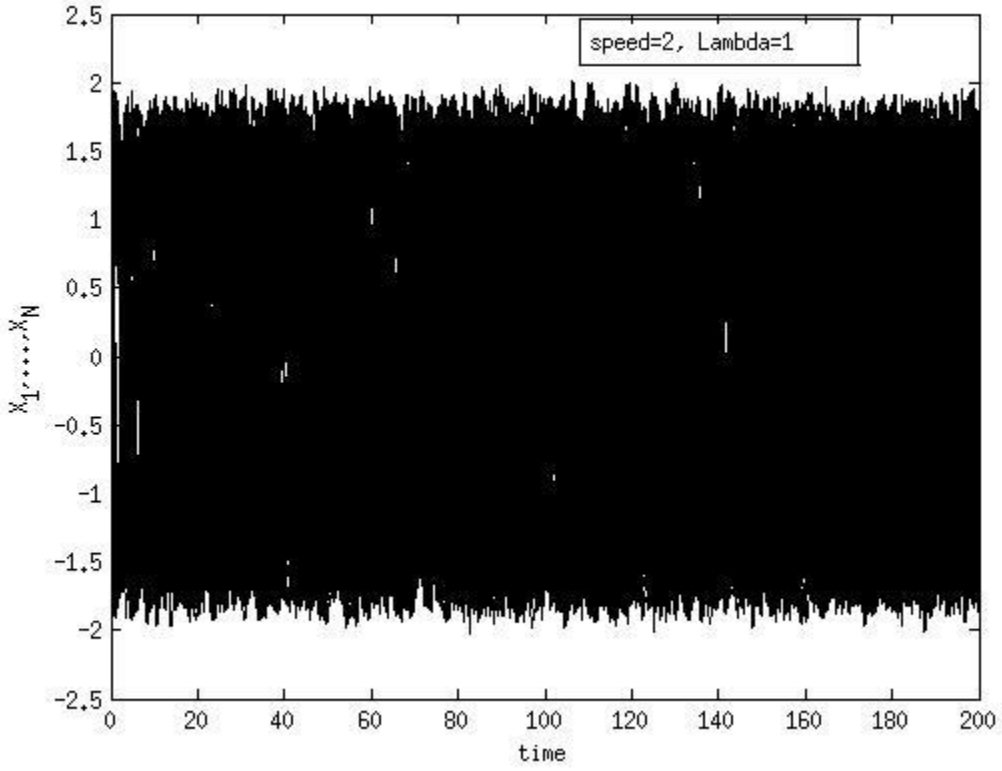

speed=30
Lambda=60
$x_1,\ldots,x_N$
Time


Lambda=1.0
$x_1,\ldots,x_2$
time

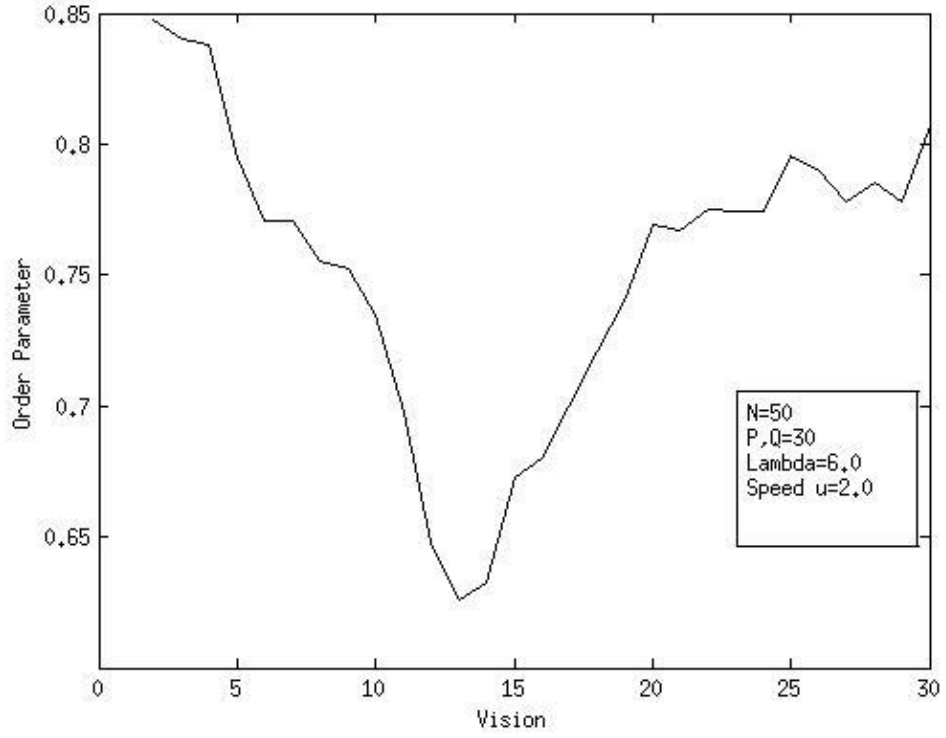

Lambda=1.0
x1
time
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
20
40
60
80
100
120
140
160
180
200

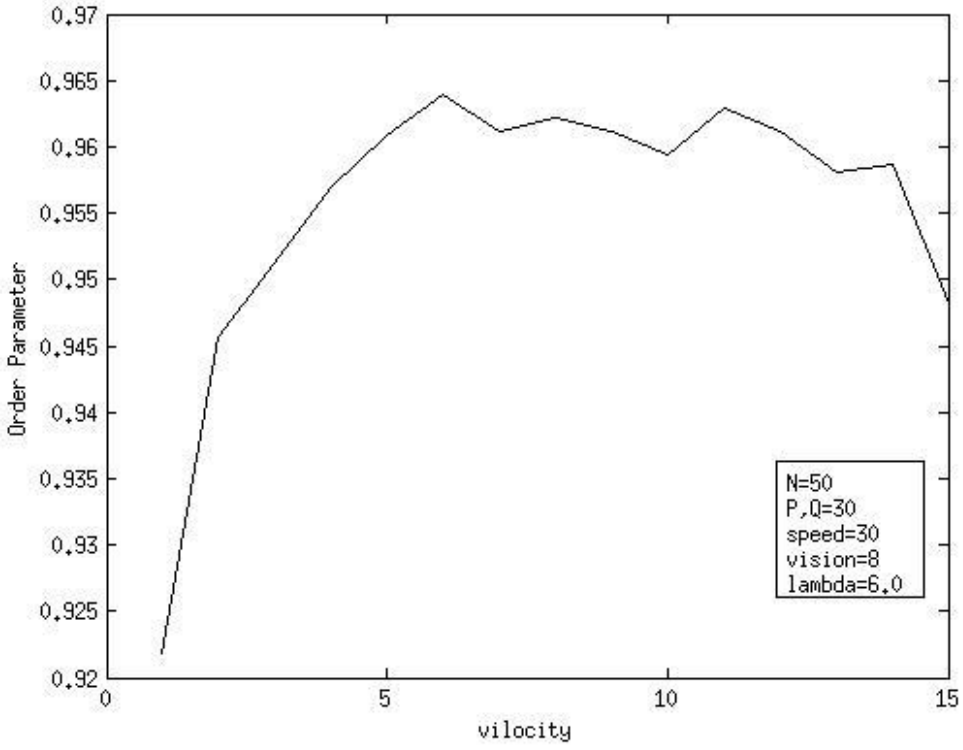

Lambda=6.0, speed=30
x1
Time
2.5
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2.5
0
20
40
60
80
100
120
140
160
180
200

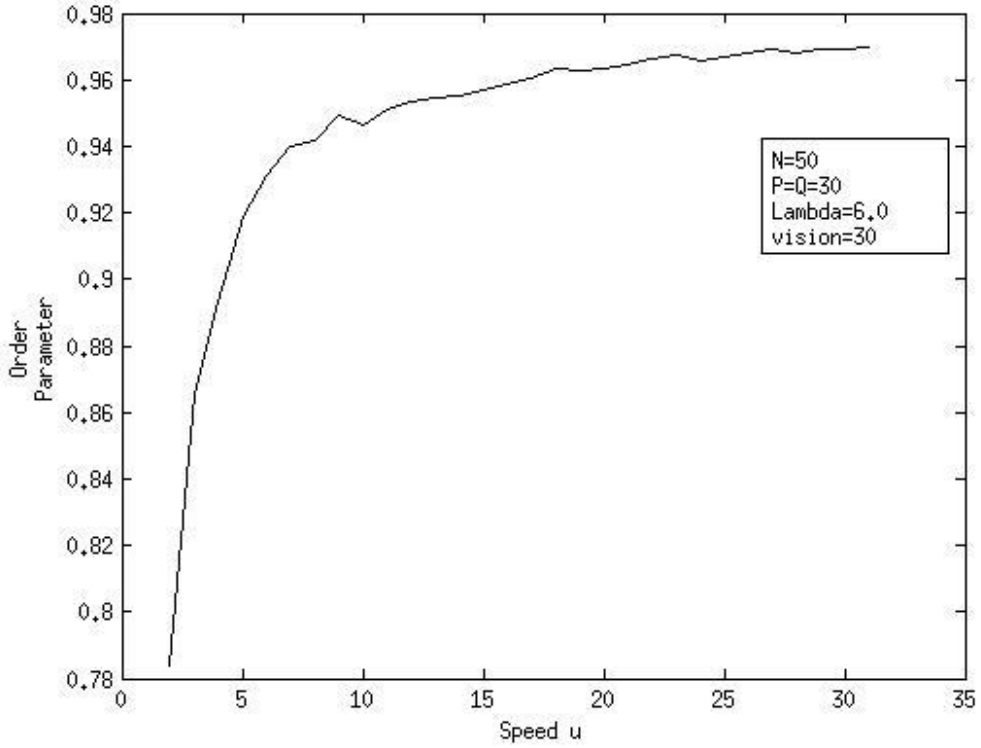


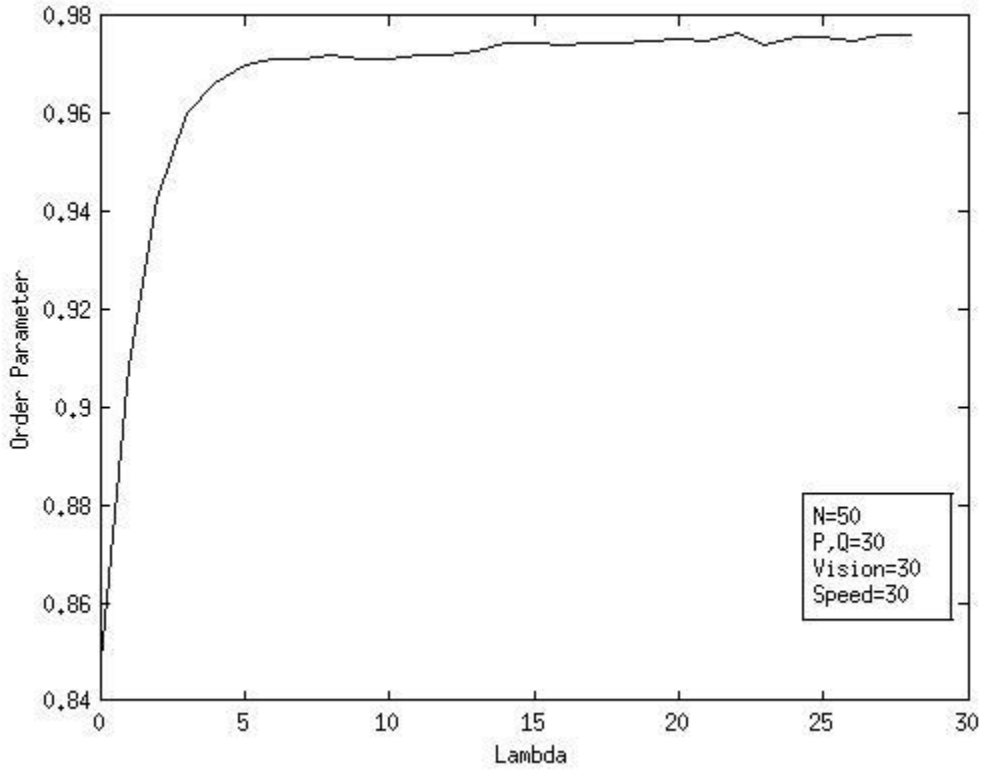



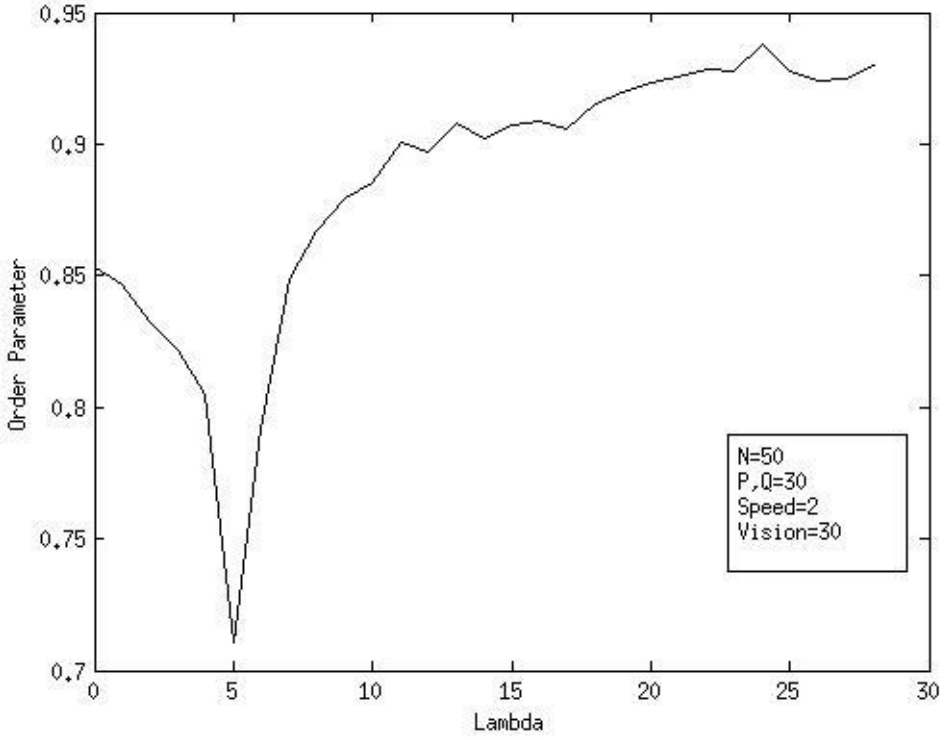

*Figures 11 - 20*

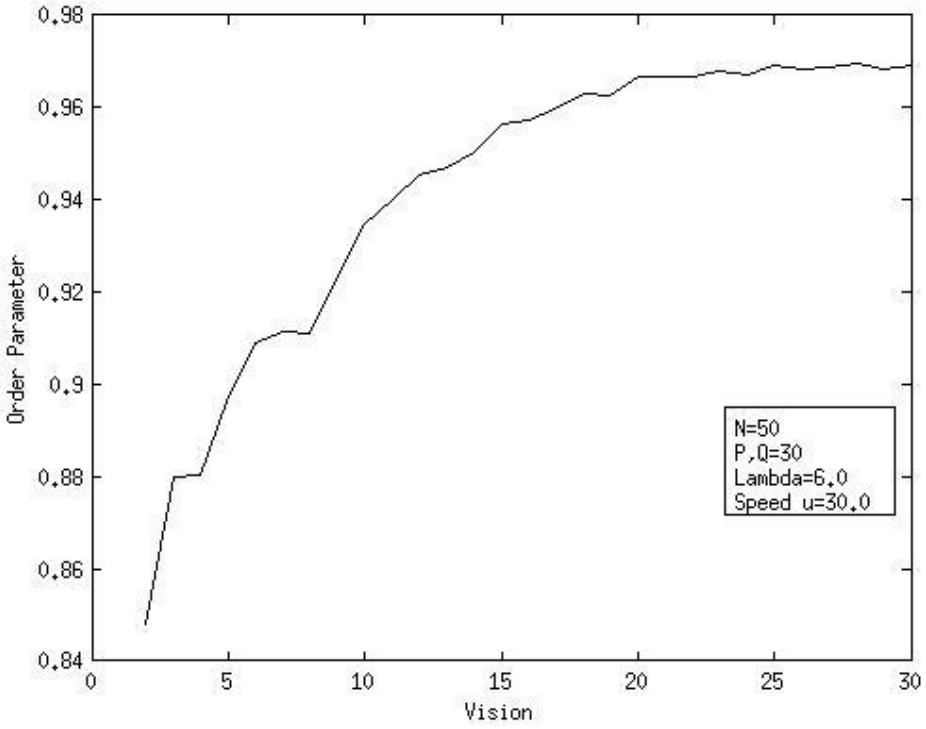

0.85
0.8
0.75
0.7
0.65
Order Parameter
0
5
10
15
20
25
30
Vision
N=50
P,Q=30
Lambda=6.0
Speed u=2.0


0.97
0.965
0.96
0.955
0.95
0.945
0.94
0.935
0.93
0.925
0.92
Order Parameter
0
5
10
15
vilocity
N=50
P,Q=30
speed=30
vision=8
lambda=6.0


0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
0.78
Order Parameter
0
5
10
15
20
25
30
35
Speed u
N=50
P=Q=30
Lambda=6.0
vision=30


0.98
0.96
0.94
0.92
0.9
0.88
0.86
0.84
Order Parameter
0
5
10
15
20
25
30
Lambda
N=50
P,Q=30
Vision=30
Speed=30




*Figures 21 – 26*

The first major difference due to the finite speed of signal propagation is that for small speed *u*, the Order parameter first decreases and then increases in oppose to the previous case where it monotonically increased. The other thing to observe is that for very small speed u and coupling strength, there is no synchronization but by increasing the speed phase synchronization is achieved. Also we see that for very large coupling as in figure 14 and figure 15 the internal motion of oscillators becomes very chaotic and erratic.

## Refrences

**[1]** Synchronization in an ensemble of spatially moving oscillators with linear and nonlinear coupling schemes - Lavneet Janagal and P. Parmananda, PHYSICAL REVIEW E 86, 056213 (2012

[2] From Phase to Lag Synchronization in Coupled Chaotic Oscillators Michael G. Rosenblum,* Arkady S. Pikovsky, and Jürgen Kurths

[3] DETERMINING LYAPUNOV EXPONENTS FROM A TIME SERIES Alan WOLF-, Jack B. SWIFT, Harry L. SWINNEY and John A. VASTANO

[4] S. Strogatz, The Emerging Science of Spontaneous Order (Hyperion, New York, 2003).

[5] S. Strogatz, Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering (Studies in Nonlinearity) (Perseus Books, Cambridge, MA, 2001).

[6] K. S. Thornburg, Jr., M. Mo¨ller, R. Roy, T. W. Carr, R. D. Li, and T. Erneux, Phys.Rev.E55, 3865 (1997).

[7] P. Ashwin, J. R. Terry, K. S. Thornburg, Jr., and R. Roy, Phys. Rev. E 58, 7186 (1998).

# Appendix

## 1. Fortran program for Lyapunov exponent

Here Lorenz system is used as an example :

```
program lorenzle
implicit none
integer :: i,j,k,l,IO=100000,NSTEP=1000000
integer, parameter :: N=3,NN=12
real :: h=0.001,t=0.0
real :: y(NN)
real, dimension(N) :: znorm,gsc,cum
! Initial cond. for nonlinear system
y(1)=10.0
y(2)=1.0
y(3)=0.0
! Initial cond. for linear system
do 10 i=N+1,NN
y(i)=0.0
10 end do
do 20 i=1,N
y((N+1)*i)=1.0
cum(i)=0.0
20 end do
open(1,file='le.dat')
write(*,*)
write(*,*)'TIME ','LAMBDA 1 ','LAMBDA 2 ','LAMBDA 3 '
do 100 i=1,NSTEP
! Integration
t=t+h
y=integrator(y)
! Construct a new orthonormal basis by GSM
! Normalize first vector
znorm(1)=0.0
do 30 j=1,N
znorm(1)=znorm(1)+y(N*j+1)**2
30 end do
znorm(1)=sqrt(znorm(1))
do 40 j=1,N
y(N*j+1)=y(N*j+1)/znorm(1)
40 end do
! Generate the new orthonormal set of vectors
do 80 j=2,N
! Generate j-1 GSR coefficients
do 50 k=1,(j-1)
gsc(k)=0.0
do l=1,N
gsc(k)=gsc(k)+y(N*l+j)*y(N*l+k)
end do
50 end do
```

```
! Construct a new vector
do 60 k=1,N
do l=1,(j-1)
y(N*k+j)=y(N*k+j)-gsc(l)*y(N*k+l)
end do
60 end do
! Calculate the vectors norm
znorm(j)=0.0
do 70 k=1,N
znorm(j)=znorm(j)+y(N*k+j)**2
70 end do
znorm(j)=sqrt(znorm(j))
! Normalize the new vector
do k=1,N
y(N*k+j)=y(N*k+j)/znorm(j)
end do
80 end do
! Update running vector magnitude
do 90 k=1,N
cum(k)=cum(k)+log(znorm(k))/log(2.0)
90 end do
! Print every IO iterations
if(mod(i,IO).eq.0) then
write(*,*)t,(cum(k)/t,k=1,N)
write(1,*)t,(cum(k)/t,k=1,N)
end if
100 end do
contains
! Function 'integrator'
function integrator(x)
real :: k(6,NN)
real, dimension(NN) :: integrator,x,x0
x0=x
k(1,:)=h*fcn(x)
x=x0+k(1,:)/4.0
k(2,:)=h*fcn(x)
x=x0+(3.0*k(1,:)/32.0)+(9.0*k(2,:)/32.0)
k(3,:)=h*fcn(x)
x=x0+(1932.0*k(1,:)/2197.0)-(7200.0*k(2,:)/2197.0)+(7296.0*k(3,:)/2197.0)
k(4,:)=h*fcn(x)
x=x0+(439.0*k(1,:)/216.0)-(8.0*k(2,:))+(3680.0*k(3,:)/513.0)-(845.0*k(4,:)/4104.0)
k(5,:)=h*fcn(x)
x=x0-(8.0*k(1,:)/27.0)+(2.0*k(2,:))-(3544.0*k(3,:)/2565.0)+(1859.0*k(4,:)/4104.0)-
(11.0*k(5,:)/40.0)
k(6,:)=h*fcn(x)
integrator=x0+(16.0*k(1,:)/135.0)+(6656.0*k(3,:)/12825.0)+(28561.0*k(4,:)/56430.0)-
(9.0*k(5,:)/50.0)+(2.0*k(6,:)/55.0)
end function integrator
! Function 'fcn'
function fcn (u)
real, dimension(NN) :: fcn,u
14
real :: si=16.0,b=4.0,R=45.92
```

```
      integer :: m
      ! LORENZ EQUATIONS
      fcn(1)= si*(u(2)-u(1))
      fcn(2)= -u(1)*u(3)+R*u(1)-u(2)
      fcn(3)= u(1)*u(2)-b*u(3)
      ! OF LINEARIZED EQUATIONS OF MOTION.
      dO m = 0,2
      fcn(4+m) = si*(u(7+m)-u(4+m))
      fcn(7+m) = (R-u(3))*u(4+m)-u(7+m)-u(1)*u(10+m)
      fcn(10+m) = u(2)*u(4+m)+u(1)*u(7+m)-b*u(10+m)
      end do
      end function fcn
      end program lorenzle
```

## 2. Program for moving oscillators with infinite speed of propagation

```
program mov_osc_ordp
  ! Internal state of oscillators ---------- 'X'
  ! Spatial position of all oscillators -------- 'S'
  ! Coordinates of oscillators in S ------------ 'SX'
  ! Orientations of oscillators----------------- 'OR'
  ! if OR(i,:)=[1,0] then orientation = +y, [-1,0] = -y, [0,1] = +x, [0,-1] = -x
  integer, parameter :: dp=kind(1.0d0)
  integer, parameter :: NN=2                  ! (Number of equations in FCN)
  integer, parameter :: N=30                  ! (Number of Oscillators)
  integer, parameter :: P=100,Q=100             ! (Size of Grid)
  integer :: vel=4       ! (Velocity and Vision)
  integer :: vision=0
  real(kind=dp) :: lambda=4.0
  integer :: i,ii,j,jj,up1,up2,uptemp
  integer :: S(P,Q),SX(N,2),OR(N,2)
  real(kind=dp), dimension(N,NN) :: UU,X,KK1,KK2,KK3,KK4,KK5,KK6
  real(kind=dp), parameter ::  pi=3.14159265359, h=0.01


  real(kind=dp) :: r1,r2,t,MU(N)
  real(kind=dp) :: sphase, cphase, order_p, m_order_p, PHASE(N)


  open(1,file='mov_oscillators.dat')
  open(2,file='orderp.dat')
  call init_random_seed()


  do 30 ii=1,20
```

```
t=0.0
m_order_p=0.0
jj=1

! Set innitial values for SX,S,X,OR,MU

! Set innitial value for S

do i=1,P
  do j=1,Q
    S(i,j)=0
  end do
end do
i=1
do while (i<=N)
  call random_number(r1)
  call random_number(r2)
  r1=1+P*r1
  r2=1+Q*r2

  if (S(int(r1),int(r2))==0) then
    S(int(r1),int(r2))=i
    i=i+1
  end if
end do

! Set innitial value for SX

do i=1,P
  do j=1,Q
    if (S(i,j)/=0) then
      SX(S(i,j),:)=[i,j]
    end if
  end do
end do

! Set innitial value for X

call random_number(X)
X=2*(X-0.5)

! Set innitial value for OR

do i=1,N
  OR(i,:)=[1,0]
end do

! Set value for MU ***Varying parameter in the function***

call random_number(MU)
MU=0.3+4.7*MU

! Start iteration
```

```
do 20 i=1,20000
  t=t+h

  ! Update positions and orientations (Random algorithm)

  do 10 j=1,N
    call random_number(r1)
    if (r1>=0.0 .and. r1<0.25) then

      up1=SX(j,1)
      up2=SX(j,2)
      uptemp=up1+vel
      if (uptemp>P) then
        uptemp=uptemp-P
      end if
      if (S(uptemp,up2)==0) then
        S(up1,up2)=0
        SX(j,1)=uptemp
        S(uptemp,up2)=j
        OR(j,:)=[1,0]
      end if

    else if (r1>=0.25 .and. r1<0.5) then

      up1=SX(j,1)
      up2=SX(j,2)
      uptemp=up1-vel
      if (uptemp<1) then
        uptemp=uptemp+P
      end if
      if (S(uptemp,up2)==0) then
        S(up1,up2)=0
        SX(j,1)=uptemp
        S(uptemp,up2)=j
        OR(j,:)=[-1,0]
      end if

    else if (r1>=0.5 .and. r1<0.75) then

      up1=SX(j,1)
      up2=SX(j,2)
      uptemp=up2+vel
      if (uptemp>Q) then
        uptemp=uptemp-Q
      end if
      if (S(up1,uptemp)==0) then
        S(up1,up2)=0
        SX(j,2)=uptemp
        S(up1,uptemp)=j
        OR(j,:)=[0,1]
      end if
```

```
        else

          up1=SX(j,1)
          up2=SX(j,2)
          uptemp=up2-vel
          if (uptemp<1) then
            uptemp=uptemp+Q
          end if
          if (S(up1,uptemp)==0) then
            S(up1,up2)=0
            SX(j,2)=uptemp
            S(up1,uptemp)=j
            OR(j,:)=[0,-1]
          end if

        end if
10      end do



      ! Integration

      UU=X

      KK1=h*FCN(UU)

      UU=X+KK1/4.0
      KK2=h*FCN(UU)

      UU=X+(3.0*KK1/32.0)+(9.0*KK2/32.0)
      KK3=h*FCN(UU)

      UU=X+(1932.0*KK1/2197.0)-(7200.0*KK2/2197.0)+(7296.0*KK3/2197.0)
      KK4=h*FCN(UU)

      UU=X+(439.0*KK1/216.0)-(8.0*KK2)+(3680.0*KK3/513.0)-(845.0*KK4/4104.0)
      KK5=h*FCN(UU)


UU=X-(8.0*KK1/27.0)+(2.0*KK2)-(3544.0*KK3/2565.0)+(1859.0*KK4/4104.0)-(11.0*KK5/40.0
)
      KK6=h*FCN(UU)


X=X+(16.0*KK1/135.0)+(6656.0*KK3/12825.0)+(28561.0*KK4/56430.0)-(9.0*KK5/50.0)+(2.0*
KK6/55.0)


      if (i>0) then
        write(1,*)t,(X(j,:),j=1,N)
      end if

      ! Calculations for order parameter
```

```
      if (i>5000) then

        do j=1,N

          sphase=atan(X(j,2)/X(j,1))

          if (X(j,1)<0.0) then
            sphase=sphase+pi
          end if

          PHASE(j)=sphase
          PHASE(j)=PHASE(j)-PHASE(1)

        end do

        sphase=0.0
        cphase=0.0

        do j=1,N
          sphase=sphase+sin(PHASE(j))
          cphase=cphase+cos(PHASE(j))
        end do

        order_p=sphase**2+cphase**2
        order_p=sqrt(order_p)/N
        m_order_p=((jj-1)*m_order_p+order_p)/jj
        jj=jj+1

      end if

20    end do

     write(*,*)vision,m_order_p
     write(2,*)vision,m_order_p
     vision=vision+1
30 end do


contains


  function FCN(VV)
    integer :: ri,rj,k,l,u1,u2,v1,v2,A1(2),A2(2),WW(2),ORW(2)
    real(kind=dp) :: VV(N,NN),FCN(N,NN),KCN,m

    ! Constants of function**********************

    !*******************************************

    ! Find oscillators in vision*************************************************

    u1=vision
```

```
u2=0.4*vision

do ri=1,N

  WW=SX(ri,:)
  ORW=OR(ri,:)
  KCN=0.0
  m=0.0

  if (ORW(1)==0) then
    A1=[0,1]
    A2=[ORW(2),0]
  else
    A1=[ORW(1),0]
    A2=[0,1]
  end if

  do k=1,u1
    do l=-u2,u2
      if (l/=0) then

        v1=WW(1)+A1(1)*k+A1(2)*l
        v2=WW(2)+A2(1)*k+A2(2)*l

        if (v1>P) then
          v1=v1-P
        else if (v1<1) then
          v1=v1+P
        end if

        if (v2>Q) then
          v2=v2-Q
        else if (v2<1) then
          v2=v2+Q
        end if

        if (S(v1,v2)/=0) then
          m=m+1
          rj=S(v1,v2)

          ! 'ri' = i th oscillator index
          ! 'rj' = j th oscillator index
          ! Coupling type*********************************************
          KCN=KCN+(VV(rj,1)-VV(ri,1))
          ! ***********************************************************

        end if
      end if
    end do
  end do
  if (m<0.5) then
    m=1.0
  end if
```

```
    ! Function*****************************************************************
    FCN(ri,1)= MU(ri)*(VV(ri,1)-(VV(ri,1)**3)/3.0 -VV(ri,2))

    ! Adding coupling term*************************
    FCN(ri,1)=FCN(ri,1)+(lambda/m)*KCN
    ! ****************************************

    FCN(ri,2)=VV(ri,1)/MU(ri)
    ! ****************************************************************************
  end do

 end function FCN

 ! ****************************************************************************
 ! Subroutine for to initialise random number seed


 subroutine init_random_seed()

 end subroutine init_random_seed


 end program mov_osc_ordp
```

## 3. Program for moving oscillators with delay coupled due to finite speed of interaction along with calculation Order Parameter

```
! Program to compute Moving-Delay-Coupled oscillators

program dc_moos_ord

  ! Internal state of oscillators ---------- 'X'
  ! Spatial position of all oscillators -------- 'S'
  ! Coordinates of oscillators in S ---------- 'SX'
 ! Orientations of oscillators---------- 'OR'
  ! if OR(i,:)=[1,0] then orientation = +y, [-1,0] = -y, [0,1] = +x, [0,-1] = -x

  integer, parameter :: dp=kind(1.0d0)
  integer, parameter :: NN=2              ! (Number of equations in FCN)
  integer, parameter :: N=50              ! (Number of Oscillators)
  integer, parameter :: P=30,Q=30         ! (Size of Grid)
  integer, parameter :: vel=1,vision=30           ! (Velocity)
  integer, parameter :: NX=N*NN

  integer :: i,j,k,l,ii,up1,up2,uptemp
  integer :: S(P,Q),SX(N,2),OR(N,2)
```

```
!integer :: vision=20                  ! (Vision)

real(kind=dp) :: h=0.01, speed=1.0, lambda=6.0
real(kind=dp) :: r1,r2,MU(N),t=0.0, time=200.0
real(kind=dp) :: X(NX), Y(1000,NX)

real(kind=dp) :: sphase, cphase, order_p, m_order_p, PHASE(N)
open(1,file='orderp.dat')

call init_random_seed()
    !************************************************************************
 do 30 ii=1,25

  t=0.0
  m_order_p=0.0
  jj=1
  ! Set innitial values for SX,S,X,OR,MU,Y,Z
  ! Set innitial value for S

  do i=1,P
    do j=1,Q
      S(i,j)=0
    end do
  end do
  i=1
  do while (i<=N)
    call random_number(r1)
    call random_number(r2)
    r1=1+P*r1
    r2=1+Q*r2

    if (S(int(r1),int(r2))==0) then
      S(int(r1),int(r2))=i

      i=i+1
    end if
  end do

  ! Set innitial value for SX

  do i=1,P
    do j=1,Q
      if (S(i,j)/=0) then
        SX(S(i,j),:)=[i,j]
      end if
    end do
  end do

  ! Set innitial value for X

  call random_number(X)
  X=2*(X-0.5)
```

```
! Set innitial value for OR

do i=1,N
  OR(i,:)=[1,0]
end do

! Set value for MU ***Varying parameter in the function***


!call random_number(MU)

MU=2.0+2.0*MU

!MU=1.0

! Innitial value for Y

do i=1,1000

  Y(i,:)=X

end do

   ! ************************************************************************


! Start iteration

do 20 iii=1,200   ! Time

  do 40 i=1,100


    ! Update positions and orientations (Random algorithm)

    if (mod(i,2)==0) then


      do 10 j=1,N

        call random_number(r1)

        if (r1>=0.0 .and. r1<0.25) then


          up1=SX(j,1)

          up2=SX(j,2)

          uptemp=up1+vel
```

```
  if (uptemp>P) then
    uptemp=uptemp-P
  end if
  if (S(uptemp,up2)==0) then
    S(up1,up2)=0
    SX(j,1)=uptemp
    S(uptemp,up2)=j
    OR(j,:)=[1,0]
  end if

else if (r1>=0.25 .and. r1<0.5) then

  up1=SX(j,1)
  up2=SX(j,2)
  uptemp=up1-vel
  if (uptemp<1) then
    uptemp=uptemp+P
  end if
  if (S(uptemp,up2)==0) then
    S(up1,up2)=0
    SX(j,1)=uptemp
    S(uptemp,up2)=j
    OR(j,:)=[-1,0]
  end if

else if (r1>=0.5 .and. r1<0.75) then
```

```
  up1=SX(j,1)
  up2=SX(j,2)
  uptemp=up2+vel
  if (uptemp>Q) then
    uptemp=uptemp-Q
  end if
  if (S(up1,uptemp)==0) then
    S(up1,up2)=0
    SX(j,2)=uptemp
    S(up1,uptemp)=j
    OR(j,:)=[0,1]
  end if

else

  up1=SX(j,1)
  up2=SX(j,2)
  uptemp=up2-vel
  if (uptemp<1) then
    uptemp=uptemp+Q
  end if
  if (S(up1,uptemp)==0) then
    S(up1,up2)=0
    SX(j,2)=uptemp
    S(up1,uptemp)=j
```

```
            OR(j,:)=[0,-1]
          end if

        end if
10      end do
      end if


      ! Integration

      X=X+h*FCN(X)

      t=t+h

      !if (i>0) then

        ! write(2,*)t,(X(j),j=1,NX)
      !end if

      do j=2,1000
        Y(j-1,:)=Y(j,:)
      end do

      do j=1,NX
        Y(1000,j)=X(j)
      end do
     ! Calculations for order parameter
```

```
if (iii>100) then

  do j=1,N

    sphase=atan(X(2*j)/X(2*j-1))

    if (X(2*j-1)<0.0) then
      sphase=sphase+pi
    end if

    PHASE(j)=sphase
    PHASE(j)=PHASE(j)-PHASE(1)

  end do

  sphase=0.0
  cphase=0.0

  do j=1,N
    sphase=sphase+sin(PHASE(j))
    cphase=cphase+cos(PHASE(j))
  end do

  order_p=sphase**2+cphase**2
  order_p=sqrt(order_p)/N
  m_order_p=((jj-1)*m_order_p+order_p)/jj
```

```
          jj=jj+1
        end if
40      end do
20    end do
      write(*,*)speed,m_order_p
      write(1,*)speed,m_order_p
      speed=speed+5
30   end do

contains

  function FCN(VV)
    integer :: jj,ri,rj,kk,ll,u1,u2,v1,v2,A1(2),A2(2),WW(2),ORW(2), dindex
    real(kind=dp) :: VV(NX),FCN(NX),KCN,m, distance, delay

    ! Constants of function**********************

    !*******************************************

    ! Find oscillators in vision*************************************************

    u1=vision
    u2=0.4*vision

    do ri=1,N

      WW=SX(ri,:)
      ORW=OR(ri,:)
```

```
KCN=0.0
m=0.0

if (ORW(1)==0) then
  A1=[0,1]
  A2=[ORW(2),0]
else
  A1=[ORW(1),0]
  A2=[0,1]
end if

do kk=1,u1
  do ll=-u2,u2
    if (ll/=0) then

      v1=WW(1)+A1(1)*kk+A1(2)*ll
      v2=WW(2)+A2(1)*kk+A2(2)*ll

      if (v1>P) then
        v1=v1-P
      else if (v1<1) then
        v1=v1+P
      end if

      if (v2>Q) then
        v2=v2-Q
```

```
        else if (v2<1) then
          v2=v2+Q
        end if

        if (S(v1,v2)/=0) then
          m=m+1
          rj=S(v1,v2)
          distance=(WW(1)-v1)**2 + (WW(2)-v2)**2
          distance=sqrt(distance)
          delay=distance/speed
          dindex=nint(delay/h)

          ! 'ri' = i th oscillator index
          ! 'rj' = j th oscillator index
          ! Coupling type**********************************************
          KCN=KCN+ Y( (1000-dindex),(2*rj-1) )- VV(2*ri-1)

          ! ************************************************************

        end if
      end if
    end do
  end do
if (m<0.5) then
  m=1.0
end if

FCN(2*ri-1)= MU(ri)*(VV(2*ri-1)-(VV(2*ri-1)**3)/3.0 -VV(2*ri))
```

```
      FCN(2*ri)=VV(2*ri-1)/MU(ri)



      ! Adding coupling term*************************

        FCN(2*ri-1)=FCN(2*ri-1)+(lambda/m)*KCN

      ! *****************************************


    end do


  end function FCN
  !*************************************************************************

 ! Subroutine for to initialise random number seed

   subroutine init_random_seed()

  end subroutine init_random_seed


end program dc_moos_ord
```